OpenJUMP um GIS 100% Java

A 10 anos um sonho hoje uma realidade

Ezequias Rocha

Um Desktop GIS 100% Java hoje é possível graças a um grupo de visionários que hoje repete o que fizeram os grandes gigantes do passado.

Há aproximadamente 3 anos, mais especificamente em 17 de Outubro de 2002, quando surgiram as primeiras idéias de um sistema de informações geográficas versátil, poderoso e ricamente portátil, não se imaginava o que em 2006 ele pudesse vir a se tornar em um sistema altamente amado por seus usuários. Entre seus fãns estão o governo Canadense, empresas de tecnologia alemã, o Japão e por que não o Brasil.

Tudo começou com uma demanda do Ministério de Ontário, mais especificamente com o departamento de Recursos Naturais da Columbia Britânica no Canadá. A demanda bancada pelo governo canadense se assemelha ao que aconteceu nos primórdios da tão conhecida ESRI, líder mundial em Geoprocessamento. Assim como na década de 80 em 2002 iniciava-se um processo sem volta. Hoje os desafios são basicamente os mesmos mas o caminho para o sucesso do OpenJUMP não será tão árduo como foi para os precursores dos Sistemas de Informação Geográfica.

A rica jornada deste bravo software começa com a solicitação para que a Vivid Solutions, empresa de software do Canadá, viesse a desenvolver um sistema GIS feito totalmente em Java e com sua interface aberta para extensões. Um núcleo enxuto e uma capacidade de importar PlugIns ainda hoje faz sucesso. Inicialmente chamado apenas de JUMP - Jump Unified Mapping Plataform (Plataforma de Mapeamento Unificada Jump) o JUMP cumpriu sua meta e após seu período de maturação seu código foi gentilmente cedido a comunidade.

Figura 1. Empresa desenvolvedora da solução JUMP

Outras versões de JUMP apareceram desde então desde o JUMP internacional (JUMP-i18n) até versões de outras empresas que moldaram-o ao seu bel prazer. O deeJUMP, o Jump da Agiles, Jump-SIGLE (francês) e o OpenJUMP, versão da comunidade. Assim como aconteceu com o OpenOffice, que adveio do StarOffice e está antenado com os padrões do mercado o OpenJUMP juntou tudo que há de mais atrativo em diversas versões sem perder a classe do projeto original.

Até o presente momento as versões não modificaram muito seu núcleo robusto baseado em três pilares fortes e distintos. Sua API é composta basicamente da JTS, da JCS e o JUMPWorkbench. Vamos descrever todas elas mostrando como tirar proveito deste conjunto de bibliotecas robustas e altamente úteis.

API JUMP	Conjunto de APIs que suportam entrada/saída de dados espaciais, feições e coleções que fundamentalmente processam algorítimos espaciais baseados no modelo geométrico
JUMPWorkbench	O JUMPWorkbench é uma interface interativa que provê um framework para a execução de funções espaciais e visualização de resultados. Ela povê muitas ferramentas de criação, visualização e manipulação do dado espacial.

A API do OpenJUMP

1. A **JTS** - Suite Topológica Jump, implementa as funcionalidade topológicas conhecidas pelos especialistas em GIS por seus padrões de **Ponto**, **Linha** e **Polígono** bem como suas inter relações (quem está contido em quem, quem toca quem, quem sobrepõe-se a quem) e seus padrões de formatação de dados proprietários e livres (Shapefile e o GML + JML).

2. A **JCS** - Suite de Conflitamento Jump, implementa toda a camada de conflitamento, informando que elementos gráficos não estão de acordo com as especificações do mercado. Ela também pode efetuar uma checagem a posteriori com base em parâmetros passados pelo usuário (como uma linha de até 100Km reais) e depois informar onde estão os conflitamentos.

3. O **JUMPWorkbench** e a interface gráfica que torna tudo isso possível ao simples clique de um mouse. Desenvolvido principalmente pelo programador canadense Jonathan Aquino, o Workbench facilita e muito a vida de quem visualiza e analisa e edita o dado geográfico diariamente.

Os desenvolvedores iniciais do projeto conclamam aos seus usuários para utilizá-lo de **3 formas** distintas e ao mesmo tempo complementares. Pode-se dizer que o JUMP e afins pode ser utilizado como uma API geográfica, como um kit de ferramentas ou como uma filosofia. Hoje o Recife implementa um projeto inovador dando visibilidade a este projeto com essas três finalidades básicas.

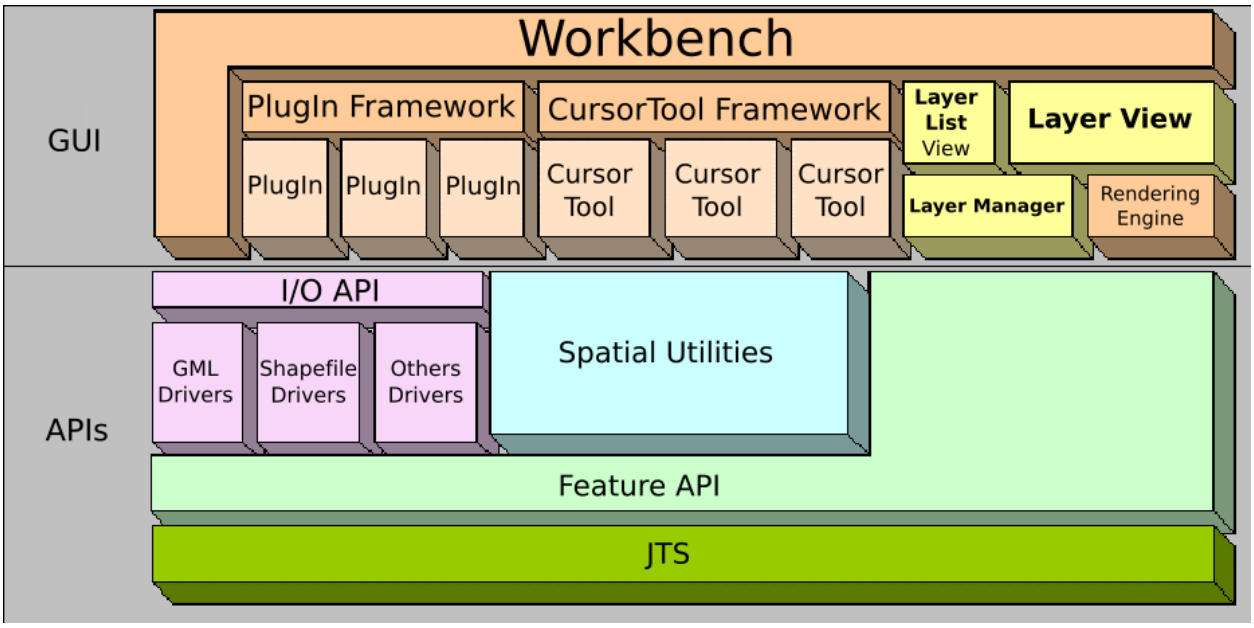

Figura 2. Framework do JUMP

O Caso do Recife

O Recife, que no final da década de 80 implementava um projeto estruturador baseado em Sistemas de Informação Geográfica sempre com softwares proprietários veio por conhecer todo o mundo do GIS livre a partir do ano de 2005. Antes disso soluções livres nunca haviam sido cogitadas. E este cenário mudou e a EMPREL - Empresa Municipal de Informática, como sempre, primou por analisar todas as possibilidades antes de continuar a enveredar por sistemas não livres.

Ao analisar-se diversos softwares livres desenvolvidos em C++, Visual Basic, e até Bibliotecas Java, a direção da empresa optou finalmente por centrar seus esforços no OpenJUMP no desktop. Por tudo que o

Java representa e por ser o nosso padrão o software foi estudado com maior afinco nos últimos meses do ano pela sua equipe. Poder estudá-lo, reprogramá-lo, extendê-lo e disseminá-lo fez os nossos olhos brilharem e ele começou a ser uma alternativa real e imediata.

Confesso que mesmo sendo aberto a novidades, fiquei um pouco receoso do real poder do software. Mesmo assim sendo nossa direção não recuou diante de tantos desafios. Foram efetuados diversos testes de performance e confiabilidade e basicamente todas as extensões foram tratadas nos testes (3D com o Java 3D, Raster GIS com o Java Advanced Image-JAI) e serviços de mapas e o acesso a bancos foram validados. O OpenJUMP não é uma simples ferramenta de visualização de dados espaciais, ele é uma realidade quer queiram ou não.

O projeto de disseminar o OpenJUMP na prefeitura do Recife (com população de mais de 1.500.000 habitantes) nos credita a informar que é o maior projeto de software livre em GIS do Brasil. O cadastro imobiliário da cidade irá conter também ferramentas beans advindos de outra API, o Geotools e irá se tornar em um marco em cadastros imobiliários no país.

Porque Java ?

Nossa empresa já conta com um know-how em desenvolvimento Java/GIS desde a primeira implementação do Cadastro Imobiliário do Recife (CI) baseada em MapObjects (uma também bem sucedida implementação Java para geoprocessamento).

Com uma comunidade altamente participativa e acolhedora, avançamos muito e acreditamos que avançaremos muito mais na implementação de novas funcionalidades para o OpenJUMP. Hoje já existe na empresa um conjunto de extensões que está sendo implementado diariamente desde o final de 2005.

A comunidade está crescendo a paços largos e uma pergunta não fica sem resposta, seja para discutir questões inerentes ao projeto seja para sanar problemas de programação. O grande número de programadores Java em relação aos programadores C e nossos cases de sucesso fizeram-nos concluir que teremos um grande e duradouro caminho a percorrer tanto como o OpenJUMP como com o Java.

Construindo um PlugIn para o JUMP

Vamos aqui criar nossa primeira extensão e ver como é fácil se incorporar à interface gráfica do sistema. Iremos criar um JInternalFrame que irá interagir com o clique do menu que será inserido ao já existente. Seu nome será "HelloWorldPlugIn.jar".

	O processo de criação de extensões (PlugIns) para o OpenJUMP é bastante simples, basta apenas empacotar seu programa Java e copiá-lo para o diretório lib/ext da instalação do software. Imediatamente o PlugIn irá ser carregado na próxima carga do sistema.

Primeiramente gostaríamos de relatar nosso expressivo agrado pela ferramenta **Eclipse** e por sua facilidade na implementação de interfaces gráficas e por sua robustez. Tanto plugins como aplicações disfarçadas de plugins foram produzidas e distribuídas utilizando-se das facilidades dele.

É importante salientar, antes de mais nada, que diversos PlugIns já foram produzidos e que no site www.OpenJUMP.org encontra-se uma gama enorme de extensões. Muitos deles estão com seu código fonte disponível e pronto para o reuso. Esta reusabilidade está nos ajudando muito aqui na empresa.

Então vamos ao código:

Listagem 1. *HelloWorldExtension.java*: extensão que irá carregar o plugin na GUI

```
package com.yourcompany;

import com.vividsolutions.jump.workbench.plugin.Extension;
import com.vividsolutions.jump.workbench.plugin.PlugInContext;

/**
 * @description
 *  - esta classe carrega o PlugIn dentro do Jump <p>
 *  - esta classe tem que ser chamada de "Extension" no fim do nome da classe
 *    para que possamos usa-lo dentro do Jump
 *
 *  @author ezequias
 */
public class HelloWorldExtension extends Extension{

    /**
     * chama o PlugIn usando o método xplugin.initialize()
     */
    public void configure(PlugInContext context) throws Exception{
        new HelloWorldPlugIn().initialize(context);
    }
}
```

Listagem 2. *HelloWorldPlugIn.java*: é o código que será executado quando o plugin for chamado no menu

```
package com.yourcompany;

import com.vividsolutions.jump.workbench.plugin.AbstractPlugIn;
import com.vividsolutions.jump.workbench.WorkbenchContext;
import com.vividsolutions.jump.workbench.plugin.EnableCheckFactory;
import com.vividsolutions.jump.workbench.plugin.MultiEnableCheck;
import com.vividsolutions.jump.workbench.plugin.PlugInContext;
import com.vividsolutions.jump.workbench.ui.plugin.FeatureInstaller;

public class HelloWorldPlugIn extends AbstractPlugIn {

      public HelloWorldPlugIn() {
            // construtor vazio
      }

      public void initialize(PlugInContext context) throws Exception {
            FeatureInstaller featureInstaller = new FeatureInstaller(context
                        .getWorkbenchContext());
            featureInstaller.addMainMenuItem(this, // exe
                        new String[] { "View" }, // menu path
                        this.getName(), // metodo .getName recebido por
```

```
                                                    // AbstractPlugIn
                        false, // checkbox
                        null, // icone
                              // enable check
                        createEnableCheck(context.getWorkbenchContext()));
        }

        public static MultiEnableCheck createEnableCheck(
                    WorkbenchContext workbenchContext) {
              EnableCheckFactory checkFactory = new EnableCheckFactory(
                        workbenchContext);

              return new MultiEnableCheck().add(checkFactory
                        .createWindowWithLayerNamePanelMustBeActiveCheck());
        }

        /**
         * Ação no item do menu: cria um novo documento a exibir
         */
        public boolean execute(PlugInContext context) throws Exception {

           context.getWorkbenchFrame().getOutputFrame().createNewDocument();
           context.getWorkbenchFrame().getOutputFrame().addText("Ola Mundo !");
           context.getWorkbenchFrame().getOutputFrame().surface();

           return true;
        }
}
```

Distribuindo nosso primeiro PlugIn

Agora vamos vê-lo em funcionamento no OpenJUMP. Utilizaremos o Eclipse por ser o que escolhemos aqui em nosso projeto. Ele tem uma ferramenta nativa de deployment bastante simples.

1º Passo: Importar Pacotes da API do JUMP

- Clique com o botão direito no Projeto que você criou no Eclipse
- Escolha o item *Java Build Path*
- Escolher a *aba Libraries*
- Clique no botão *Add External JARs...*
- Selecionar os jars na pasta lib onde você instalou o OpenJUMP previamente
- Clique em *OK*

2º Passo: Distribuindo o PlugIn

Depois de escrever o código conforme as listagens 1 e 2, citadas você pode exportá-lo como um ".jar" clicando com o botão direito em Project1 e escolhendo a opção `Export...`

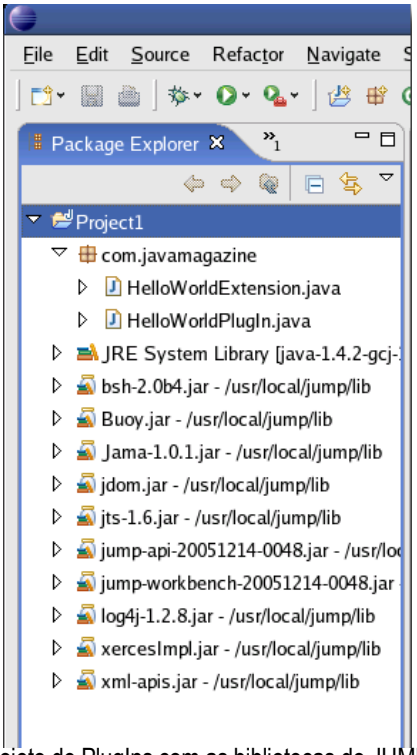

Figura 3. Seu projeto de PlugIns com as bibliotecas do JUMP

Após a escolha por exportar você deve dizer ao Eclipse que formado será exportado seu código. Escolha a opção *JAR file* e clique em *Next*

Verifique se todas as classes que existem em nosso PlugIn estão selecionadas na caixa de checagem a esquerda no item *Select the resources to export:* e passe para o último passo.

Ainda na mesma janela, só que agora no item *JAR file:* defina em que diretório irá salvar o seu PlugIn.

No nosso caso a exportação vai parar no diretório onde o OpenJUMP foi instalado. Fizemos desta forma para que você já veja, logo ao iniciar o OpenJUMP se seu plugin foi distribuido corretamente.

Agora clique no botão Finish para concluir a exportação e terminar o processo de criação do nosso primeiro PlugIn.

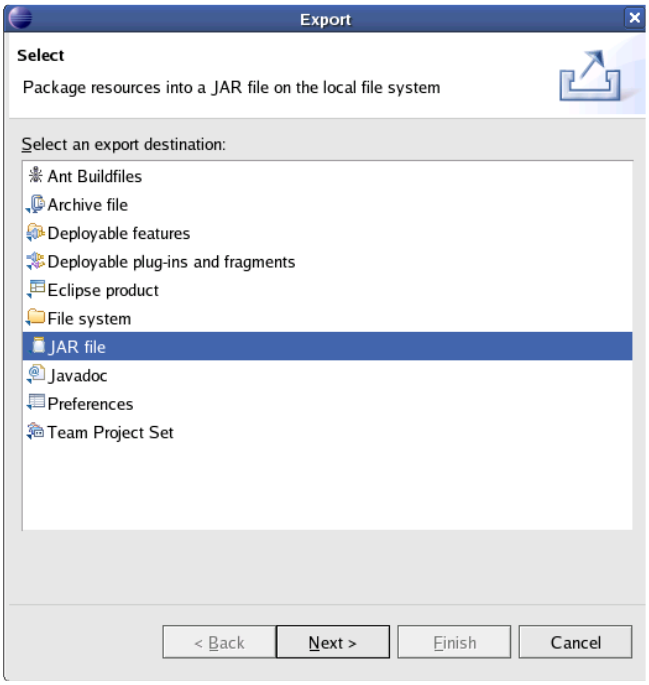

Figura 4. Selecione JAR file para ir para o último passo antes de testá-lo

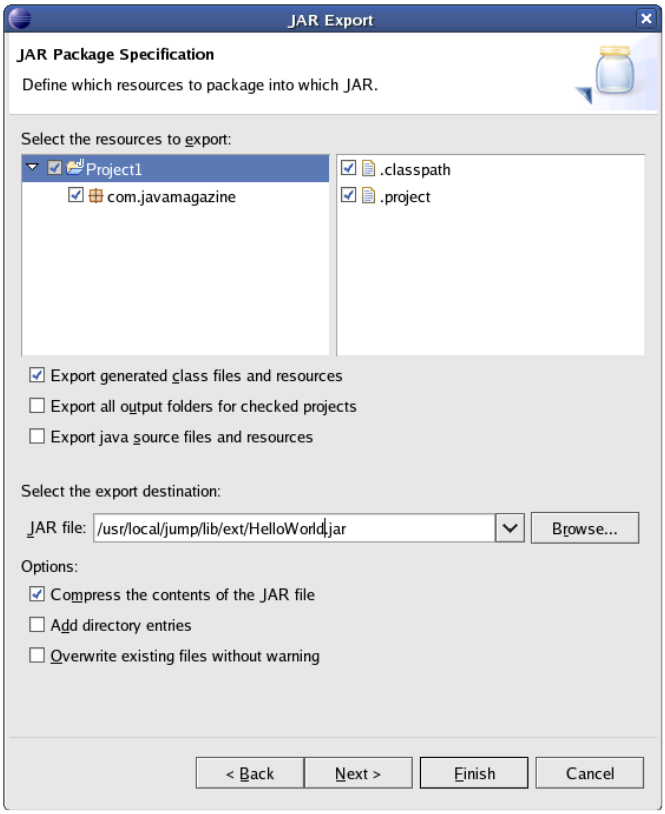

Figura 5. Defina aqui que classes serão exportadas e onde salvar seu PlugIn

3º Passo: Testando-o

Agora que o PlugIn foi criado e instalado, podemos agora ir a passo final colocando o OpenJUMP em ação.

Vamos iniciá-lo ?

Antes disto vamos tomar cuidado para que ele fique no idioma inglês setando o parâmetro de internacionalização *-i18n en* no script de inicialização do programa (JUMPWorkbench.bat ou JUMPWorkbench-unix.sh), na pasta \bin, com um editor de texto.

Agora sim inicie o OpenJUMP. Em sua inicialização você o verá sendo carregado na tela splash.

Você verá um menu *View* um item a mais chamado *HelloWorldPlugIn* que irá abrir um JInternalFrame boa sorte programadores, todos são bem vindos no projeto. Muitas das suas dúvidas podem ser tiradas na lista de discussão do JUMP ou na Wiki page do projeto.

Características gerais do OpenJUMP

Vamos listar aqui algumas das características gerais alcançadas pelo projeto. Dentre muitas de suas funcionalidades podemos destacar:

- Capacidade de abrir e editar dados geográficos no formato shapefile (formato amplamente utilizado hoje)
- Capacidade de abrir e gerar arquivos geográficos em conforme com os padrões OGC
- Fácil exibição, manipulação e exportação de dados geográficos
- Leitura de bases no PostgreSQL/PostGIS
- Ferramentas de análise espacial como União, Interseção, Sobreposição, Translação etc
- Leitura de imagens de satélite (Raster com o arquivo worldfile através de PlugIn)
- Edição de mapas e de atributos relacionados a ele
- Ligação Mapa atributo e atributo mapa
- Ferramentas de navegação (zoom/pan)
- Capacidade de ler serviços de Servidores de Mapas WMS e WFS (com PlugIn)
- Vasta biblioteca de PlugIns

Veremos agora a interface gráfica do projeto.

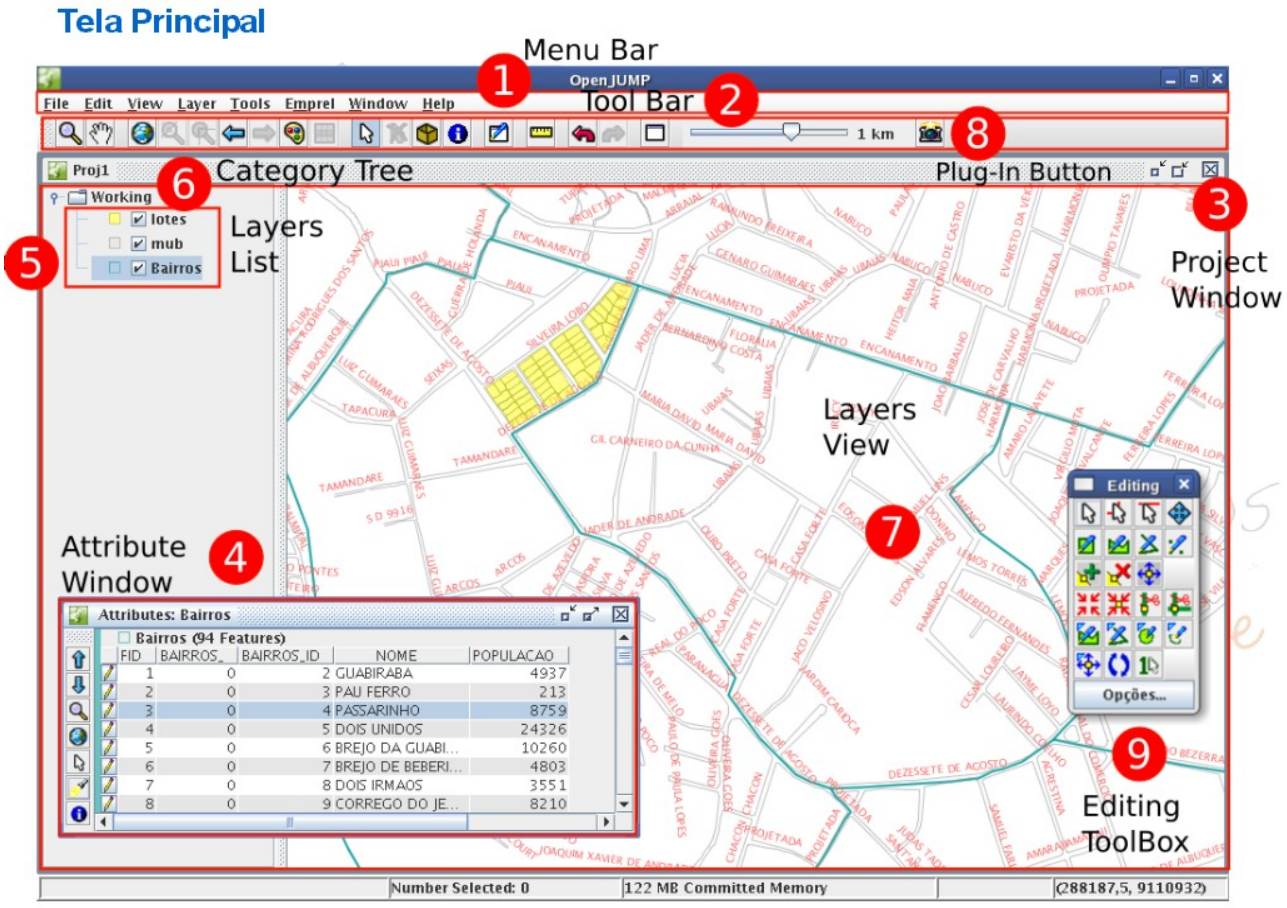

Figura 6. Nível atual da interface gráfica do OpenJUMP

O OpenJUMP em ação

Veremos aqui algumas telas do OpenJUMP acessando dados raster e vetoriais.

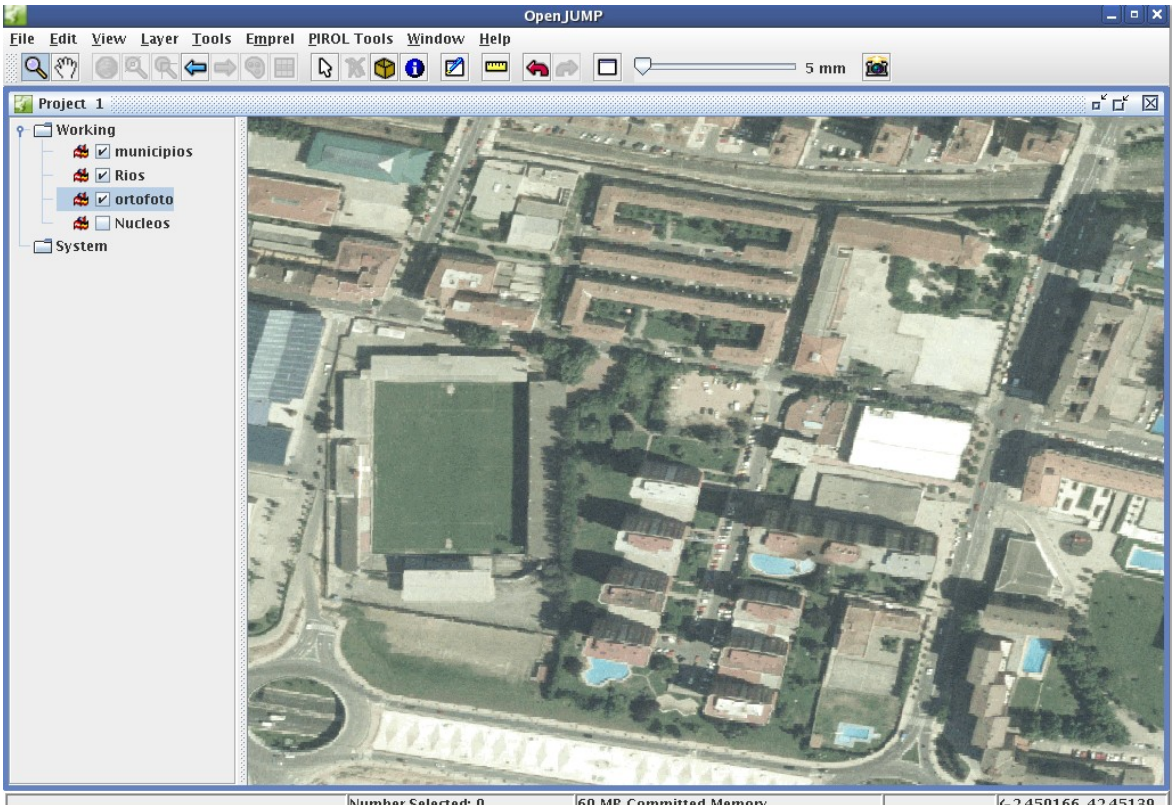

Figura 7. Aqui abrindo um serviço de mapas remoto com fotografias aéreas

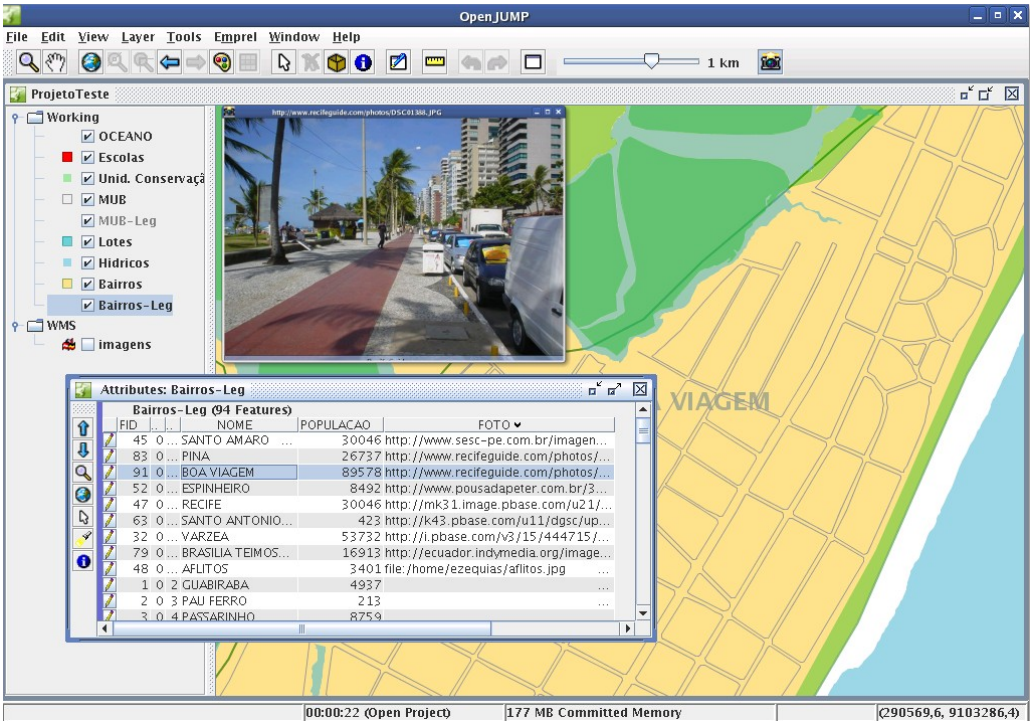

Figura 8. Aqui abrindo uma imagem associada ao mapa [PlugIn genuinamente nacional]

Links

www.openjump.org
Wiki page do Projeto
www.jump-project.org
Website do Projeto Inicial (O JUMP)
www.vividsolutions.com
Empresa desenvolvedora do JUMP

Ezequias Rodrigues da Rocha (ezequias@recife.pe.gov.br, www.recife.pe.gov.com) é Analista de Sistemas, trabalha com geoprocessamento desde 1998 quando elaborou o primeiro trabalho com GIS aplicado à Defesa Civil. Trabalha hoje na EMPREL - Empresa Municipal de Informática, vinculada à prefeitura do Recife, com sistemas livres aplicados à Sistemas de Informação Geográfica.